**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: Использование указателей в языке программирования СИ

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Любимов В.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

## Цель работы.

Изучить работу с указателями, многомерные массивы и строковые массивы и принципы работы с ними в языке программирования СИ.

## Задание.

Напишите программу, которая форматирует некоторый текст и выводит результат на консоль.

На вход программе подается текст, который заканчивается предложением "Dragon flew away!".

Предложение (кроме последнего) может заканчиваться на:

* . (точка)
* ; (точка с запятой)
* ? (вопросительный знак)

Программа должна изменить и вывести текст следующим образом:

* Каждое предложение должно начинаться с новой строки.
* Табуляция в начале предложения должна быть удалена.
* Все предложения, которые заканчиваются на '?' должны быть удалены.
* Текст должен заканчиваться фразой "Количество предложений до n и количество предложений после m", где n - количество предложений в изначальном тексте (без учета терминального предложения "Dragon flew away!") и m - количество предложений в отформатированном тексте (без учета предложения про количество из данного пункта).

## Выполнение работы.

1. *i, j –* счётчики в циклах
2. *size –* размер буфера
3. *sent –* указатель на строку
4. *ch –* считанный символ
5. *content –* массив указателей на строки
6. *cp –* считанная строка
7. *oldlen –* изначальное количество предложений
8. *text.i* – количество предложений в данный момент времени
9. *text.size* – размер буфера, выделенного под указатели на строки
10. *text.content* – массив указателей на строки, то есть обрабатываемый текст
11. *mytext* – тип данных, представляющий собой структуру *MyText*

Функция char\* *InputSentence()* возвращает строку, считанную посимвольно. Табуляции и пробелы, идущие перед первым буквенным символом строки , не считываются. Строка заканчивается символом конца предложения и нулевым символом.

Функция *struct MyText InputText()* возвращает структуру *MyText*, имеющую поля *i* – количество предложений в данный момент времени, *text.size* – размер буфера, выделенного под указатели на строки, *text.content* – массив указателей на строки, то есть обрабатываемый текст. Текст считывается по предложениям , пока не будет считано терминальное предложение.

Функция *struct MyText NewLine(mytext text)* возвращает структуру *MyText.* В конец каждого предложения добавляется символ переноса строки.

Функция *struct MyText Shift(mytext text, int start)* возвращает структуру *MyText.* Начиная суказателя с индексом *start,* сдвигает все следующие указатели на единицу влево. Уменьшает на единицу *text.i.*

Функция *struct MyText ExcludeSentence(mytext text, int oldlen)* возвращает структуру *MyText.* Удаляет все предложения оканчивающиеся на «?». Применяет функцию *struct MyText Shift(mytext text, int start).*

Функция *void OutputText(mytext text, int oldlen)*  выводит на экран текст построчно. После терминального предложения добавляет предложение о количестве предложений в тексте, как требуется в задании.

Функция *void MyFree(mytext text)* очищает выделенную в ходе выполнения программы память. Использует функцию стандартной библиотеки *free.*

## Выводы.

Были изучены принципы работы c указателями и массивами в языке СИ.

Разработана программа, выполняющая считывание исходных данных и обрабатывающая их, как требует задание. Для обработки входных данных были созданы структура для хранения текста и функции обработки текста, использующие эту структуру. Обработка исключительных ситуаций не производится, ибо гарантирована корректность входных данных и наличие терминального предложения.

# Приложение А Исходный код программы

Название файла: main.с

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define LEN 10

struct MyText{

char \*\*content;

int size;

int i;

};

char\* InputSentence(){

int size = LEN;

int i = 0;

char\* sent = malloc(size \* sizeof(char));

char ch;

do{

ch = fgetc(stdin);

if (!((ch == ' ' || ch == '\t')&& i == 0)){

sent[i] = ch;

i++;

};

if(i == size - 3){

size = size + LEN;

sent = realloc(sent, size \* sizeof(char));

};

}while (!strchr("!.;?", ch));

sent[i] = '\0';

return sent;

}

struct MyText InputText(){

int size = LEN;

int i = 0;

char\*\* content = malloc(size \* sizeof(char\*));

while(1){

char \*cp = InputSentence();

if(strstr(cp, "Dragon flew away!")){

content[i] = cp;

i = i + 1;

break;

}

else{

content[i] = cp;

i++;

if(i == size - 2){

size = size + LEN;

content = realloc(content, size \* sizeof(char\*));

};

};

};

struct MyText text;

text.content = content;

text.size = size;

text.i = i;

return text;

}

typedef struct MyText mytext;

struct MyText NewLine(mytext text){

int i = 0;

int j = 0;

while (i < text.i){

j = 0;

while(!strchr("!.;?", text.content[i][j])){

j = j + 1;

};

text.content[i][j + 1] = '\n';

text.content[i][j + 2] = '\0';

i = i + 1;

};

return text;

}

struct MyText Shift(mytext text, int start){

int i = start;

while (i < (text.i - 1)){

text.content[i] = text.content[i+1];

i = i + 1;

};

text.i = text.i - 1;

return text;

}

struct MyText ExcludeSentence(mytext text, int oldlen){

int i = 0;

int shift = 0;

while (i < text.i){

if (text.content[i][strlen(text.content[i])-2] == '?'){

text = Shift(text, i);

i = i - 1;

};

i = i + 1;

};

return text;

}

void OutputText(mytext text, int oldlen){

int i = 0;

while (i < text.i){

printf("%s", text.content[i]);

i = i + 1;

};

printf("%s %d %s %d", "Количество предложений до", oldlen - 1, "и количество предложений после", text.i - 1);

}

void MyFree(mytext text){

int i = 0;

while(i < text.i){

free(text.content[i]);

i = i + 1;

};

free(text.content);

}

int main(){

mytext text;

text = InputText();

int oldlen = text.i;

text = NewLine(text);

text = ExcludeSentence(text, oldlen);

OutputText(text, oldlen);

MyFree(text);

return 0;

}